# Testul de număr prim C++

ENUNT: Se citește de la tastatură un număr n. Determinați dacă numărul citit este prim și afișați un mesaj corespunzător.

Codul afișat mai jos determină dacă numărul are divizori proprii, pe care îi numără folosind variabila nr. În final, dacĂ numărul are divizori proprii, programul afișează mesajul “Numărul nu este prim”, iar daca nr = 0 (numărul nu are divizori proprii), afișează mesajul “Numărul este prim”.
Un număr este prim dacă are doar divizori improprii (dacă este divizibil doar cu 1 și cu el însuși).

```
#include <iostream>
using namespace std;
int n,d,nr;

int main()

{

cin>>n;
for(d=2;d<=n/2;d++)
      if(n%d==0) nr++;

if (n<=1) nr++;

if(nr==0) cout<<"Numarul este prim";
else cout<<"Numarul NU este prim";

return 0;
}
```

# VARIANTA OPTIMA

# -orice numar (daca nu e prim) are cel putin un divizor mai mic sau egal cu radical din el

Exemplu: pentru nr 14 gasim divizor un numar <=radical(14)- 2<=3

```
#include <iostream>
using namespace std;
int n,d,nr;
int main()
{
cin>>n;
d=2;
while (d*d<=n && n%d!=0) d++;

if(n>=2 && d*d>n) cout<<"Numarul este prim";
else cout<<"Numarul NU este prim";

return 0;
}
```

# Descompunerea în factori primi C++

ENUNT: Se citește de la tastatură un număr n. Determinați și afișați pe ecran descompunerea în factori primi ai acestuia.

```
#include <iostream>
int main()
{
int n,d,p;
cin>>n;
d=2;
while(n>1)
{
	p=0;
	while(n%d==0)
	{
	p++;
	n=n/d;
	}
	if(p>0) cout<<d<<" la puterea "<<p<<endl;
	d=d+1;
	}
}
```

OBS. Putem îmbunătăți algoritmul (eficientiza) astfel: pentru numerele prime (pentru care nu există nici un divizor în afară de 1 și numărul însuși) în cazul în care se depășește radical din număr putem ”sări” la numărul însuși.

```
#include <iostream>
int main()
{
int n,d,p;
cin>>n;
d=2;
while(n>1)
{
	p=0;
	while(n%d==0)
	{
	p++;
	n=n/d;
	}
	if(p>0) cout<<d<<" la puterea "<<p<<endl;
	if (d*d<=n) d=d+1;
	else d=n;
}
}
```

# Divizorii proprii ai unui număr C++

ENUNT: Se citește de la tastatură un număr n. Determinați și afișați divizorii proprii ai acestuia în caz că există.
Codul afișat mai jos parcurge intervalul [2, n/2] printr-o instrucțiune for, folosind variabila d. În cazul în care aceasta divide numărul n, atunci d este afișată, întrucât este un divizor propriu al numărului n.

Un număr n poate avea divizori proprii și improprii.
Divizorii improprii ai unui număr sunt 1 și numărul însuși.
Divizorii proprii sunt restul numerelor care divid pe n, cuprinse între [n, n/2].

```
#include <iostream>
using namespace std;
int main()

{ int n,d;
      cin>>n;
      cout<<"Divizorii proprii ai lui n sunt: ";
      for(d=2;d<=n/2;d++)
            if(n%d==0)  cout<<d<<" ";
return 0;
}
```

# Cel mai mare divizor comun (cmmdc) C++

ENUNT: Se citesc de la tastatură două numere a și b. Determinați și afișați pe ecran cel mai mare divizor comun al acestora.

Codul de mai jos determină cel mai mare divizor comun al lui a și b prin scăderi repetate, afișându-l la final.

## PRIN SCĂDERI REPETATE

```
#include <iostream>
using namespace std;
int main()
{
      int a,b;
      cin>>a>>b;

      //se determina cel mai mic divizor comun prin scaderi repetate
      while(a!=b)
            if(a>b) a=a-b;
            else b=b-a;

      //se afiseaza a (care memoreaza acum cmmdc-ul dintre cele 2 numere)
      cout<<"Cmmdc = "<<a;
}
```

## PRIN ÎMPĂRȚIRI REPETATE

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,r;
    cin >>a>>b;
    r=a%b;
    while (r!=0)
    {
        a=b;
        b=r;
        r=a%b;
    }
    cout << "cmmdc=" << b<<endl;
    // b este ultimul rest nenul (acesta este cmmdc)
    return 0;
}
```

***OBS. Algoritmul de determinare al cmmdc-ului prin împărțiri repetate este mai rapid decât cel prin scăderi repetate***

# Cel mai mic multiplu comun (cmmmc) C++

ENUNT: Se citesc de la tastatură doua numere a și b. Determinați și afișați cel mai mic multiplu comun al acestora.
Codul de mai jos află cel mai mare divizor comun (cmmdc) al celor două numere pentru ca mai apoi să-l folosească pentru a determina cel mai mic multiplu comun (cmmmc) folosind formula consacrată,
cmmmc = (a*b) / cmmdc.

```
#include <iostream>
using namespace std;
int main()
{
	int a,b,x,y;
	cin>>a>>b;
	// păstrăm în 2 variabile auxiliare (copii) cele 2 numere, deoarece
	la finalul algoritmului de determinare cmmdc, aceste variabile își
	pierd valorile inițiale)
	x=a;
	y=b;
	while(x!=y)
		if(x>y)x=x-y;
		else y=y-x;
	cout<<"cmmmc = "<<(a*b)/x;
}
```